AI Magazine Volume 23 Number 3 (2002) (© AAAI)

Support Vector Machines and Kernel Methods

The New Generation of Learning Machines

Nello Cristianini and Bernhard Schölkopf

■ Kernel methods, a new generation of learning algorithms, utilize techniques from optimization, statistics, and functional analysis to achieve maximal generality, flexibility, and performance. These algorithms are different from earlier techniques used in machine learning in many respects: For example, they are explicitly based on a theoretical model of learning rather than on loose analogies with natural learning systems or other heuristics. They come with theoretical guarantees about their performance and have a modular design that makes it possible to separately implement and analyze their components. They are not affected by the problem of local minima because their training amounts to convex optimization. In the last decade, a sizable community of theoreticians and practitioners has formed around these methods, and a number of practical applications have been realized. Although the research is not concluded, already now kernel methods are considered the state of the art in several machine learning tasks. Their ease of use, theoretical appeal, and remarkable performance have made them the system of choice for many learning problems. Successful applications range from text categorization to handwriting recognition to classification of gene-expression data.

In many respects, the last few years have witnessed a paradigm shift in the area of machine learning, comparable to the one of the mid-1980s when the nearly simultaneous introduction of decision trees and neural network algorithms revolutionized the practice of pattern recognition and data mining. In just a few years, a new community has gathered, involving several thousands of researchers and engineers, a yearly workshop, web sites, and textbooks. The focus of their research: support vector machines (SVMs) and kernel methods.

Such paradigm shifts are not unheard of in the field of machine learning. Dating back at least to Alan Turing's famous article in *Mind* in 1950, this discipline has grown and changed with time. It has gradually become a standard piece of computer science and even of software engineering, invoked in situations where an explicit model of the data is not available but, instead, we are given many training examples. Such is the case, for example, of handwriting recognition or gene finding in biosequences.

The nonlinear revolution of the 1980s, initiated when the introduction of back-propagation networks and decision trees opened the possibility of efficiently learning nonlinear decision rules, deeply influenced the evolution of many fields, and paved the way for the creation of entire disciplines, such as data mining and a significant part of bioinformatics. Until then, most data analysis was performed with linear methods essentially based on the systems developed in the 1960s, such as the PERCEPTRON. The asymmetry however remained: A number of optimal algorithms and theoretical results were available for learning linear depen-

dencies from data, but for nonlinear ones, greedy algorithms, local minima, and heuristic searches were all that was known. In a way, researchers had come to accept that the theoretical elegance and practical convenience of linear systems was not achievable in the more powerful setting of nonlinear rules.

A decade later, however, kernel methods made it possible to deal with nonlinear rules in a principled yet efficient fashion, which is why we identify them with a new generation, following the linear learning machines, one in the 1960s and the first generation of nonlinear ones in the 1980s. The consequences of this new revolution could be even more far reaching. In many ways, kernel methods represent an evolution of the subsymbolic learning approaches such as neural networks, but in other ways, they are an entirely new family of algorithms and have more in common with statistical methods than with classical AI.

The differences with the previous approaches are worth mentioning. Most of the learning algorithms proposed in the past 20 years had been based to a large extent on heuristics or on loose analogies with natural learning systems, such as the concept of evolution, or models of nervous systems. They were mostly the result of creativity and extensive tuning by the designer, and the underlying reasons for their performance were not fully understood. A large part of the work was devoted to designing heuristics to avoid local minima in the hypothesis-search process. The new pattern-recognition algorithms overcome many such limitations by using a number of mathematical tools.

To start with, in the last decade, a general theory of learning machines has emerged and with it the possibility of analyzing existing algorithms and designing new ones to explicitly maximize their chances of success. The impact of learning theory on machine learning has been profound, and its effects have also been felt in the industrial applications. Furthermore—and somehow independently—new efficient representations of nonlinear functions have been discovered and used for the design of learning algorithms. This representation makes use of so-called "kernel functions," discussed later, and has a number of useful properties.

The combination of these two elements has led to powerful algorithms, whose training often amounts to convex optimization. In other words, they are free from local minima. This use of convex optimization theory, largely a consequence of the novel representation, marks a radical departure from previous greedy search algorithms and, furthermore, enables researchers to analyze general formal properties of the learning system's output.

In a way, researchers now have the power of nonlinear function learning together with the conceptual and computational convenience that was, to this point, a characteristic of linear systems.

SVMs are probably the best-known example of this class of algorithms. Introduced in 1992 at the Conference on Computational Learning Theory (Boser, Guyon, and Vapnik 1992), it has since been studied, greatly generalized, and applied to a number of different problems. The general class of algorithms resulting from this process is known as kernel methods or kernel machines. They exploit the mathematical techniques mentioned earlier to achieve the maximal flexibility, generality, and performance, both in terms of generalization and in terms of computational cost. They owe their name to one of the central concepts in their design: the notion of kernel functions, used in the representation of the nonlinear relations discovered in the data and discussed later.

The growing impact of this new approach in the larger field of machine learning can be gauged by looking at the number of researchers and events related to it. The research community gathered around these algorithms is very diverse, including people from classic machine learning, neural networks, statistics, optimization, and functional analysis. It meets at a yearly workshop, held at the Neural Information Processing Systems Conference for the past five years. Since the first such workshop, the growth of this field has been rapid: Textbooks have appeared (Cristianini and Shawe-Taylor 2000; Schölkopf and Smola 2002); many hundreds of papers have been published on this topic; and all the major conferences and journals in machine learning, neural networks, and pattern recognition have devoted increasing attention to it, with special issues, dedicated sessions, and tutorials. The recently launched *Journal of Machine Learning Research* has a regular section for kernel methods.[1] An increasing number of universities teach courses entirely or partly dedicated to kernel machines.

Currently, SVMs hold records in performance benchmarks for handwritten digit recognition, text categorization, information retrieval, and time-series prediction and have become routine tools in the analysis of DNA microarray data.

In this article, we survey the main concepts in the theory of SVMs and kernel methods; their statistical and algorithmic foundations; and their application to several problems, such as text categorization, machine vision, handwriting recognition, and computational biology.

Statistical Learning Theory

The kind of learning algorithm that we are talking about can mathematically be described as a system that receives data (or observations) as input and outputs a function that can be used to predict some features of future data. In other words, it automatically builds a model of the data being observed and exploits it to make predictions. Generalization is the activity of inferring from specific examples a general rule, which also applies to new examples. Of course, building a model capable of generalizing requires detecting and exploiting regularities (or patterns) in the data.

For example, a learning system can be trained to recognize a handwritten digit 5 from the other nine classes of digits. Such a system would initially be presented with a number of images of handwritten digits and their correct class *label,* and after learning, it would be required to correctly label new, unseen images of handwritten digits.

Statistical learning theory (Vapnik 1998) models this as a function estimation problem: The function in this case maps representations of images to their correct class label. Given some observations of this function at random positions (the training set of labeled images), it requires the hypothesis to correctly label new examples with high accuracy. The performance in predicting labels in a test set of images is known as *generalization performance*. Vladimir Vapnik and his coworkers at the Institute of Control Sciences of the Russian Academy of Science in Moscow pioneered this approach to statistical pattern recognition, and since then, many other researchers refined their early results.

It turns out that the risk of a learned function making wrong predictions depends on both its performance on the training set and a measure of its complexity. In other words, it is not sufficient to accurately describe the training data, but it is necessary to do so with a suitably simple hypothesis (an idea related to the well-known Occam's razor principle). For example, it is always possible to interpolate 5 points in the plane with a polynomial of, say, degree 25, but nobody expects the resulting function to have any predictive power; however, we are more likely to trust predictions of a linear function interpolating them.

In Vapnik's theory, the key observation is that the complexity of a function is not an absolute concept, but it depends on the class of functions from which it was extracted. Complexity of a learning machine is measured by counting the number of possible data sets that the learning machine could perfectly explain without errors using elements of its associated function class. Now suppose the learning machine has explained one data set at hand. Certainly, if the learning machine's capacity is high enough such that it can explain all possible data sets, then it will come as no surprise if it can explain the given one. Thus, from the mathematical point of view, no generalization to new data points can be guaranteed. If, however, the learning machine explains the given data although its capacity is small, then we have reason to believe that it will also work well on new data points that it has not seen during training.

This insight is formalized in statistical learning theory, and it leads to bounds on the risk of the learned hypothesis making wrong predictions that depend on many factors, including the training sample size and the complexity of the hypothesis (see sidebar 1).

The theory identifies which mathematical characteristics of learning machines control this quantity. Named after its inventors Vapnik and Chervonenkis, the most famous such characteristic is called the VC dimension. More refined notions of complexity (or capacity) have been proposed in recent years.

The resulting bounds can point algorithm designers to those features of the system that should be controlled to improve generalization. In many classical learning algorithms, the main such feature is the dimensionality of the function class or data representation being used. The class of algorithms described in this article, however, exploit very high dimensional spaces, and for this purpose, it is crucial to obtain bounds that are not affected by it. One of such bounds is at the basis of the support vector machine algorithm and is a reason why this algorithm does not suffer from the "curse of dimensionality."

Although these general ideas had been around (but, to some extent, neglected) since the 1960s, the crucial development for practitioners occurred in the 1990s. It turned out that not only did this theory explain the success of several existing learning procedures, but more importantly, it enabled researchers to design entirely new learning algorithms, not motivated by any analogy, just aimed at directly optimizing the generalization performance.

Support Vector Machines

In introducing the main ideas of SVMs, our reasoning is informal, confining mathematical arguments to separate sidebars and pointing the interested readers to the references at the end of the article.

The fundamental idea of kernel methods is

to embed the data into a vector space, where linear algebra and geometry can be performed. One of the simplest operations one can perform in such space is to construct a linear separation between two classes of points. Other operations can involve clustering the data or organizing them in other ways. The use of linear machines is easier if the data are embedded in a high dimensional space.

However, it is important to embed the data into the correct space, so that the sought-after regularities can easily be detected, for example, by means of linear separators. Informally, one would like "similar" data items to be represented by nearby points in the embedding space. High dimensional spaces are often used for this purpose.

Two major problems occur when pursuing this line of action. First, on the *statistical side,* if the dimensionality of such a space is very high, then it is trivial to find a separation consistent with the labels in the training data. Hence, according to the learning-theoretic reasoning sketched earlier, one cannot expect this trivial separation to predict well the labels of new unseen points. For this reason, we have to incorporate ideas from statistical learning theory to rule out meaningless explanations of the data, which would lead to overfitting. We see later how this is performed in practice.

Second, on the *computational side,* working directly in very high dimensional spaces can be demanding, and this cost can pose a severe limit on the size of problems that can be solved with this algorithm. This problem is addressed in a rather clever way by using kernel functions, as follows.

Kernels are inner products in some space but not necessarily in the space where the input data come from. They can often be computed efficiently. By reformulating the learning algorithms in a way that only requires knowledge of the inner product between points in the embedding space, one can use kernels without explicitly performing such embedding. In other words, rather than writing the coordinates of each point in the embedding space, we directly compute the inner products between each pair of points in such space. This is sometimes called *implicit mapping* or *implicit embedding*. If we call ϕ the embedding function, then a kernel can be written as $k(x, z) = \langle \phi(x), \phi(z) \rangle$. Surprisingly, many learning algorithms can be formulated in this special way, that is, kernelized. Among these are many standard linear discriminant techniques, clustering procedures, regression methods such as ridge regression, and principal components analysis (PCA).

The functions learned by kernel machines can be represented as linear combinations of kernels computed at the training points:

$$f(x) = \sum y_i \alpha_i k(x_i, x) + b$$

In approximation theory and functional analysis, this representation was already used, although never combined with statistical learning theory results and pattern-recognition algorithms such as the ones described here. Importantly, they correspond to linear functions in the embedding space, and hence, powerful linear methods and analysis can be applied.

The basic algorithm we analyze deals with the problems of learning classifications into two categories. SVMs achieve this by first embedding the data into a suitable space and then separating the two classes with a hyperplane (see sidebar 2). Results from statistical learning theory show that the generalization power of a hyperplane depends on its margin, defined as the distance from the nearest data point, rather than the dimensionality of the embedding space. This result provides the motivation for the following simple algorithm:

> Embed the data into a high dimensional space and then separate it with a maximum margin hyperplane. Because the margin measures the distance of the closest training point to the hyperplane, the maximum margin hyperplane not only correctly classifies all points, it actually does so with the largest possible "safety margin."

The decision function learned by such a machine has the form

$$f(x) = sign(\sum_i \alpha_i y_i k(x_i, x) + b)$$

where x_i, y_i, k are training points, their labels, and the kernel function, respectively, and x is a generic test point.

It is important to note that the problem of constructing the maximal margin hyperplane reduces to a quadratic programming problem, whose convex objective function can always be maximized efficiently under the given constraints. The absence of the so-called local minima in the cost function marks a radical departure from the standard situations in systems such as neural networks and decision trees—to cite the most popular ones—that were forced to utilize greedy algorithms to find locally optimal solutions. An entire set of empirical tricks—that often absorbed most of the researchers' attention—can thus be replaced by a well-developed field of mathematics: convex optimization.

SVMs can always be trained to the optimal

solution in polynomial time, which is one of the reasons for their fast uptake among practitioners in pattern recognition. However, other important features contribute to their performance.

With convex optimization concepts, it is possible to show that in the solution, only some of the α coefficients are nonzero, namely, the ones lying nearest to the separation hyperplane. The other points could be removed from the training set without loss of information about the classification function. Thus, what is called a *sparseness* of the solution is induced, which is the key for many efficient algorithmic techniques for optimization as well as an analysis of SVM generalization performance based on the concept of data compression. Such points are called *support vectors* (hence the name of the entire approach).

Another major property of this class of algorithms is their modularity: The maximal margin algorithm is independent of the kernel being used, and vice versa. Domain knowledge and other techniques dictate the choice of kernel function, after which the same margin-maximizing module is used. It is important to note that the maximal margin hyperplane in the embedding space will usually correspond to a nonlinear boundary in the input space, as illustrated by figure 1.

The high generalization power of large margin classifiers is the result of the explicit optimization of statistical bounds controlling the risk of "overfitting." The choice of the kernel function is the only major design choice. However, this choice can be guided by domain expertise, keeping in mind that a kernel can be regarded as a similarity measure and that often domain experts already know effective similarity measures between data points. Such has been the case, for example, in the field of text categorization, where a kernel inspired by information-retrieval techniques has successfully been used (Joachims 1998). Domain-specific kernels are also increasingly used in bioinformatics.

The maximal margin hyperplane is, however, not the only classification algorithm to be used in combination with kernels. For example, the well-known Fisher discriminant procedure and methods motivated by the Bayesian approach to statistics and machine learning have been combined with kernels, leading to other interesting nonlinear algorithms whose training often amounts to convex optimization.

Regression

Similarly, several kernel-based methods for regression have been proposed. For example, the classic ridge regression algorithm acquires

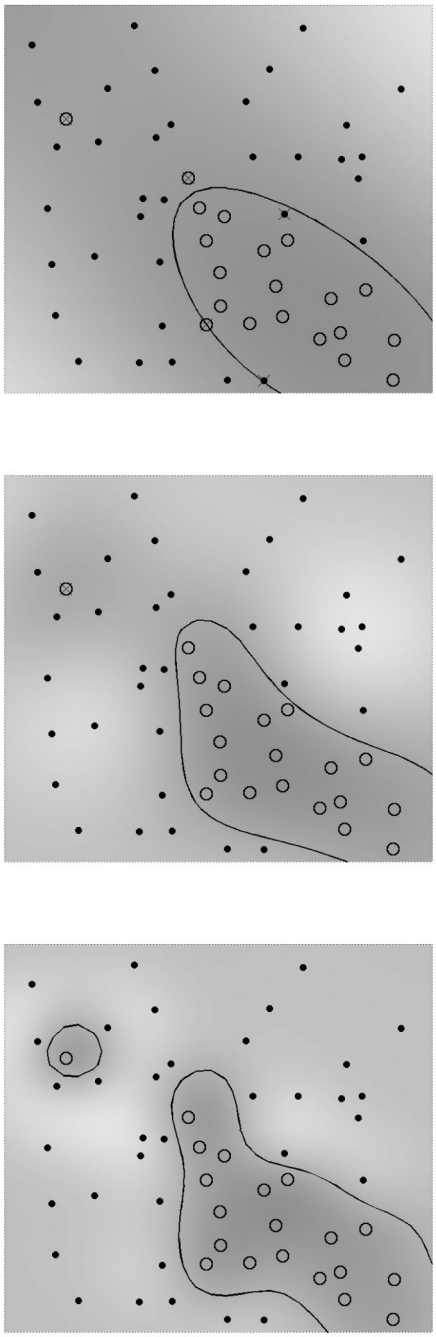


Figure 1. A Simple Two-Dimensional Classification Problem (Separate Balls from Circles), with Three Possible Solutions of Increasing Complexity.

The first solution, chosen from a class of functions with low capacity, misclassifies even some rather simple points. The last solution, however, gets all points right but is so complex that it might not work well on test points because it is basically just memorizing the training points. The medium-capacity solution represents a compromise between the other two. It disregards the outlier in the top left corner but correctly classifies all other points.

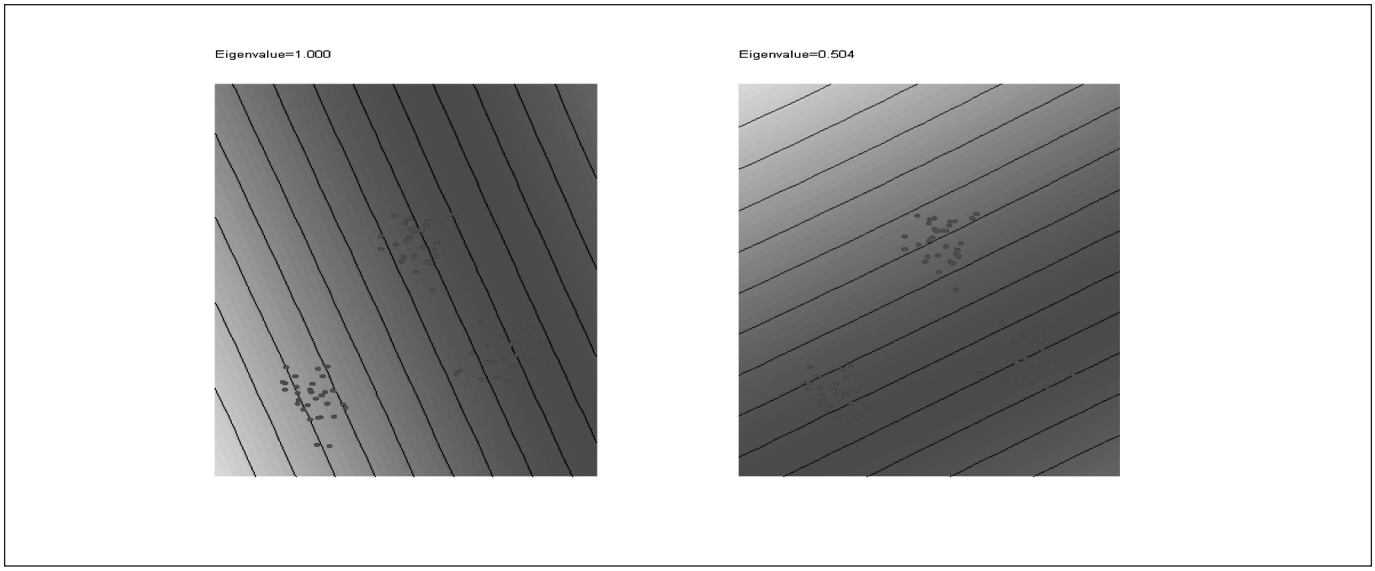


Figure 2. The First (and Only) Two Principal Components of a Two-Dimensional Toy Data Set.

Shown are the feature-extraction functions, that is, contour plots of the projections onto the first two eigenvectors of the covariance matrix. Being a linear technique, principal component analysis cannot capture the nonlinear structure in the data set.

a completely new flavor when executed in a nonlinear feature space. Interestingly, the resulting algorithm is identical to a well-known statistical procedure called Gaussian processes regression and to a method known in geostatistics as Krieging.

Also in this case, the nonlinear regression function is found as a linear combination of kernels by solving a convex optimization problem. A special choice of loss function also leads to sparseness and an algorithm known as SVM regression.

Unsupervised Learning

The same principles and techniques deployed in the case of SVMs have been exported to a number of other machine learning tasks. Unsupervised learning deals with the problem of discovering significant structures in data sets without an external signal (such as labels provided by human expert).

For example, PCA is a technique aimed at discovering a maximally informative set of coordinates in the input space to obtain a more efficient representation of the data. This technique was routinely performed in a linear fashion, using as coordinate basis vectors the eigen vectors of the data set's covariance matrix (roughly speaking, the directions in which the data vary most). Several tricks existed to overcome the obvious limitations of linear systems, but the discovery that this algorithm can be used in any kernel-induced embedding space opened the possibility to perform nonlinear PCA in a principled way, now known as *kernel PCA* (figures 2, 3, and 4).

Novelty detection is the task of spotting anomalous observations, that is, points that appear to be different from the others. It can be used, for example, in credit card fraud detection, spotting of unusual transactions of a customer, and engine fault detection and medical diagnosis. An efficient SVM method for representing nonlinear regions of the input space where normal data should lie makes use of kernel embedding.

Clustering is the task of detecting *clusters* within a data set, that is, sets of points that are "similar to each other and different from all the others." Obviously, performing clustering requires a notion of similarity, which has usually been a distance in the data space. Being able to reformulate this problem in a generic embedding space created by a nonlinear kernel has paved the way to more general versions of clustering.

Applications

The number of successful applications of this class of methods has been growing steadily in the last few years. The record for performance in text categorization, handwriting recognition, and some genomics applications is currently held by SVMs. Furthermore, in many other domains, SVMs deliver performance comparable to the best system, requiring just a minimum amount of tuning.

We review just a few cases: text categoriza-

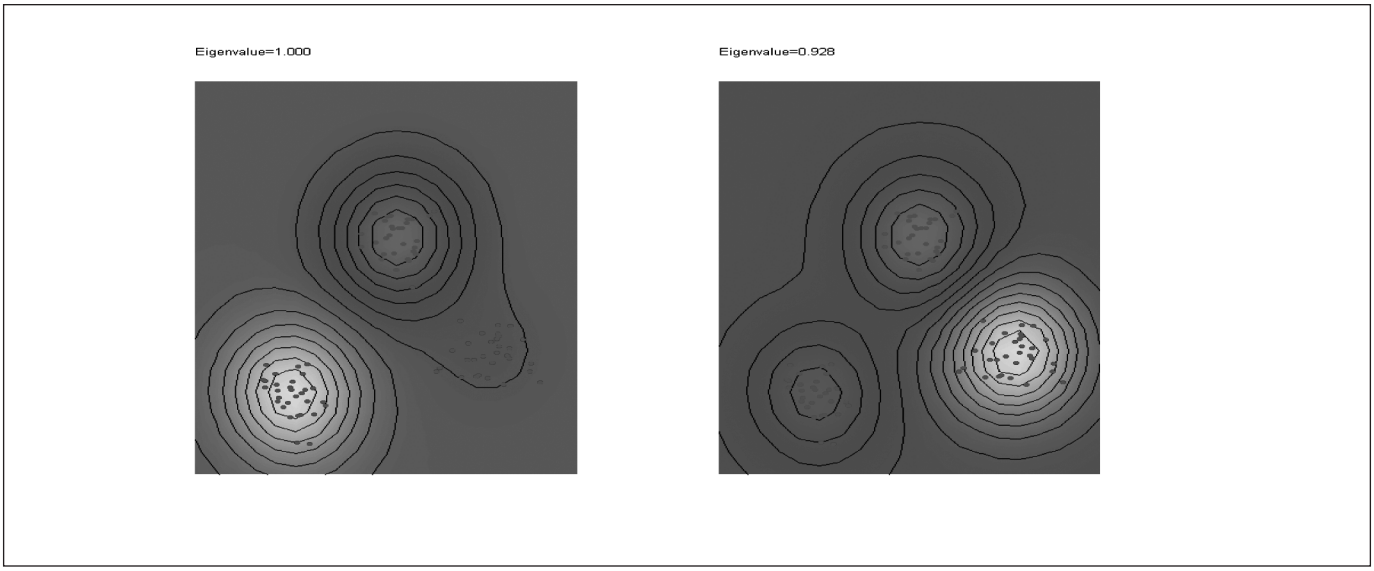


Figure 3. The First Two Principal Components for Kernel Principal Components Analysis (PCA), That Is, PCA Performed in the Feature Space Associated with a Kernel Function.

The two feature extractors identify the cluster structure in the data set: The first one (left) separates the bottom left cluster from the top one; the second one (right) then disregards this distinction and focuses on separating these two clusters from the third one.

tion, handwritten digit recognition, and gene expression data classification. We briefly summarize here the results of these benchmarks but strongly recommend readers access the original papers to obtain a full description and justification of the cost functions being used for the comparison, the data sets, the experimental design, and other fundamental statistical considerations that we could not address here because of space limitations. A proper discussion would require several pages for each experiment.

Text Categorization

Researchers have constructed a kernel from a representation of text documents known in information retrieval as a *bag of words*. In this representation, a document is associated with a sparse vector that has as many dimensions as there are entries in a dictionary. Words that are present in the document are associated with a positive entry, whereas words that are absent will have a zero entry in the vector. The weight given to each word depends also on its information content within the given corpus. The similarity between two documents is measured by the cosine between vectors, which corresponds to a kernel.

The combination of this kernel with SVMs has created the most efficient algorithm for the classification of news wire stories from Reuters, a standard benchmark.

Comparisons with four conventional algorithms (*k*-nearest neighbor, the decision tree learner C4.5, the standard Rocchio algorithm, and naïve Bayes) reported by Thorsten Joachims (1998) on two standard benchmarks (the Reuters and the Oshumed corpora) show SVMs consistently outperforming these methods. Performance was measured by microaveraged precision/recall breakeven point across 10 categories for Reuters and 23 categories for Oshumed. In particular, for the Reuters data, the average performance across the 10 categories ranges for the conventional methods from 72 to 82 where SVMs achieve 86.0 with polynomial kernels and 86.4 with Gaussian kernels. For the Oshumed data, the conventional systems ranged from 50.0 to 59.1, but SVMs achieved 65.9 with polynomial kernels and 66.0 with Gaussian kernels.

Since then, SVMs have been used in a number of text categorization applications.

Handwritten-Digit Recognition

During the 1990s, the so-called MNIST set emerged as the "gold standard" benchmark for pattern-recognition systems in what was then the Adaptive Systems Research Group at AT&T Bell Labs. This group pioneered the industrial use of machine learning as well as contributed fundamental insights to the development of the field in general. The MNIST training set contains 60,000 handwritten digits, scanned in a resolution of 28 x 28 pixels. All major algorithms were tested on this set, and a sophisticated multilayer neural net (called LeNet) had long held the benchmark record. Recently, an

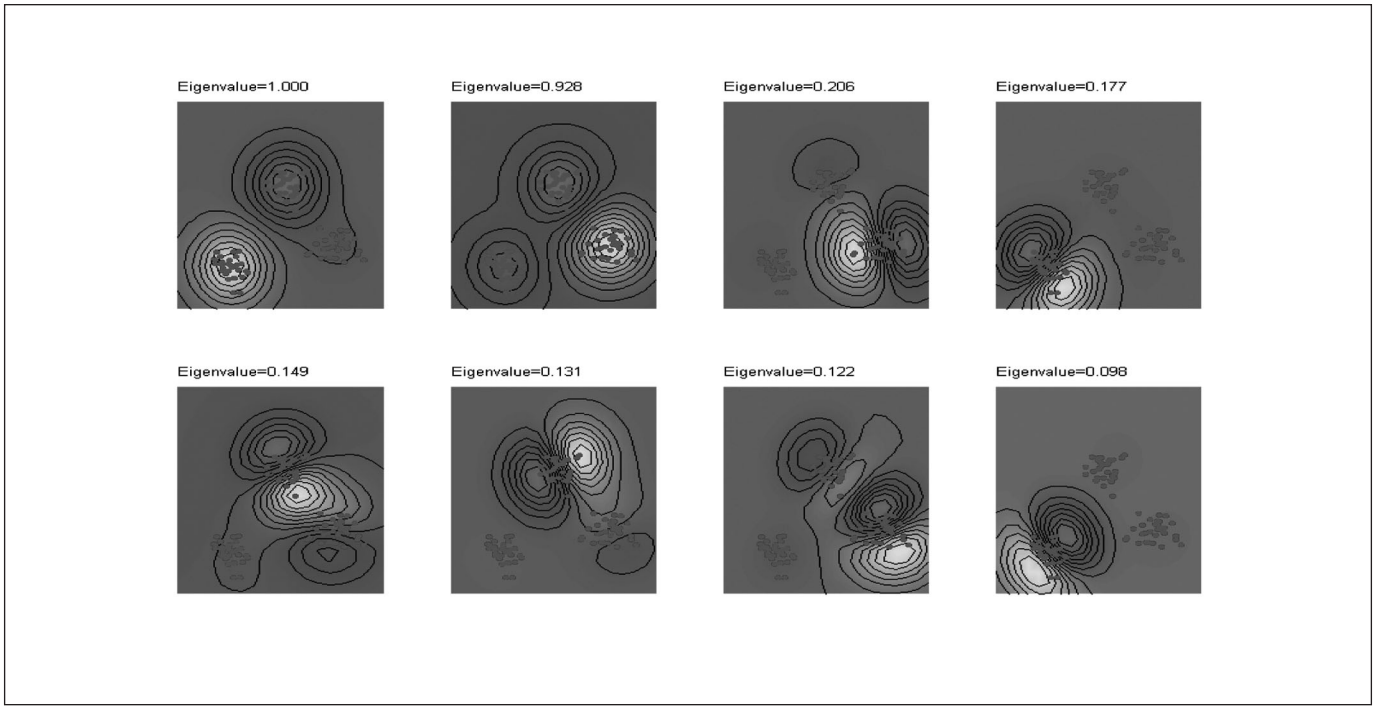


Figure 4. Because It Is Performed in a High-Dimensional Feature Space, Kernel Principal Components Analysis Can Extract More Than Just the Two Feature Extractors Shown in Figure 2.

It turns out that the higher-order components analyze the internal structure of the clusters; for example, component number 4 (top right) splits the bottom left cluster in a way that is orthogonal to component number 8 (bottom right).

SVM invariant to a class of image transformations has achieved a new record result, thus taking over one of the previous strongholds of neural nets. The previous record was owned by a boosted version of the sophisticated neural network LeNet4, which has been refined for years by the ATT Labs, with 0.7-percent test error. SVMs with so-called jittered kernels (DeCoste and Schoelkopf 2002) achieved 0.56 percent. On the same data set, 3 nearest neighbors obtains 2.4 percent, 2-layer neural networks 1.6 percent, and SVMs without specialized kernels 1.4 percent, illustrating the importance of kernel design.

Gene Expression Data

Modern biology deals with data mining as much as it does with biochemical reactions. The huge mass of data generated first by the Genome Project and then by the many postgenomic techniques such as DNA microarrays calls for new methods to analyze data. Nowadays, the problem of obtaining biological information lies just as much in the data analysis as it lies in the development of actual measuring devices.

However, the high dimensionality and the extreme properties of bioinformatics data sets represent a challenge for most machine learning systems. In particular, gene-expression data generated by DNA microarrays are high dimensional and noisy and usually expensive to obtain (hence only available in small data sets). A typical microarray experiment produces a several-thousand–dimension vector, and its cost means that usually only a few dozen such vectors are produced.

SVMs are trained on microarray data to accurately recognize tissues that are, for example, cancerous or to detect the function of genes and, thus, for genomic annotation.

SVMs were first used for gene-function prediction based on expression data at the University of California at Santa Cruz in 1999 (Brown et al. 2000). The same group later pioneered the use of SVMs for cancer diagnosis based on tissue samples. In both cases, they delivered state-of-the-art performance, and since this time, these experiments were repeated and extended by a number of groups in essentially all the leading institutions for bioinformatics research. Kernel methods are now routinely used to analyze such data.

In this first experiment, SVMs were compared to four conventional algorithms (decision tree learner C4.5, the perceptron decision tree learner MOC1, Parzen windows, and the Fisher discriminant) on a data set of about

Learning Classifications with Large Margin Hyperplanes

In pattern classification, the objective is to estimate a function f: $\Re^n \to \{-1, +1\}$ using training data—that is, n-dimensional vectors x_i and class labels y_i such that f will correctly classify new examples (x, y) with high probability. More precisely, we would like the predicted label $f(x)$ to equal the true label y for examples (x, y), which were generated from the same underlying probability distribution $P(x, y)$ as the training data.

If we put no restriction on the class of functions that we choose our estimate f from, however, even a function that does well on the training data—for example, by satisfying $f(x_i) = y_i$ for all i—need not generalize well to unseen examples. Suppose we have no additional information about f (for example, about its smoothness). Then the values on the training patterns carry no information whatsoever about values on novel patterns. Hence, learning is impossible, and minimizing the training error does not imply a small expected test error.

Statistical learning theory shows that it is crucial to restrict the class of functions that the learning machine can implement to one with a capacity that is suitable for the amount of available training data.

We call *error* of the function f the probability of mislabeling a point x whose label is y:

$$\_ = P(\{x \mid f(x) \neq y\})$$

and the theory aims at upper bounding this quantity with an expression that includes observable characteristics of the learning machine.

For example, the probability of mislabeling a new point from the same distribution, with a function that perfectly labels the training sample, increases with a measure of the function complexity, known as the VC dimension (but other measures of capacity exist). Controlling the capacity is, hence, a crucial step in the design of a learning machine: Given two functions with identically performing training sets, the one with lower capacity has the higher probability of generalizing correctly on new points.

It follows from the previous considerations that to design effective learning algorithms, we must come up with a class of functions whose capacity can be computed. The algorithm should then attempt to keep the capacity low and also fit the training data. Support vector classifiers are based on the class of hyperplanes

$\langle w, x\rangle + b = 0$

$w, x \in \Re^n, b \in \Re$

corresponding to decision functions

$f(x) = sign(\langle w, x\rangle + b)$

It is possible to prove that the *optimal hyperplane*, defined as the one with the maximal margin of separation between the two classes (figure A), stems from the function class with the lowest capacity. This hyperplane can be constructed uniquely by solving a constrained quadratic optimization problem whose solution w has an expansion in terms of a subset of training patterns that lie closest to the boundary (figure A). These training patterns, called *support vectors,* carry all relevant information about the classification problem. Omitting many details of the calculations, there is just one crucial property of the algorithm that we need to emphasize: Both the quadratic programming problem and the final decision function depend only on dot products between patterns, which is precisely what lets us generalize to the nonlinear case.

The key to set up the optimization problem of finding a maximal margin hyperplane is to observe that for two points $x+$ and $x-$ that lie nearest to it, it is true that

$$\langle w, x^+\rangle + b = +1$$

$$\langle w, x^-\rangle + b = -1$$

$$\langle w, (x^+ - x^-)\rangle = 2$$

$$\left\langle \frac{w}{\|w\|}, (x^+ - x^-)\right\rangle = \frac{2}{\|w\|}$$

Hence, the margin is inversely proportional to the norm of w. Therefore, we need to minimize the norm of w subject to the given constraints:

$\min\langle w, w\rangle$

$s.t. y_i[\langle w, x_i\rangle + b] \geq 1$

This equation can be solved by constructing a Lagrangian function:

$$\frac{1}{2}\langle w, w\rangle - \sum \alpha_i \left[y_i\left(\langle w, x_i\rangle + b\right) - 1\right]$$

$$\alpha_i \geq 0$$

where the coefficient α is a Lagrange multiplier, and by transforming it into the corresponding dual Lagrangian by imposing the optimal conditions,

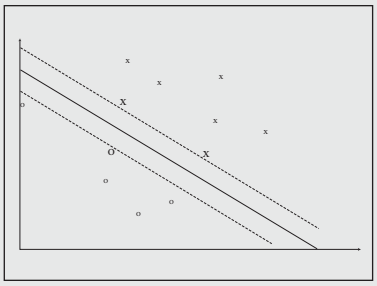


Figure A. A Maximal Margin Separating Hyperplane.
Notice that its position is determined by the nearest points, called *support vectors*.

$$\frac{\partial L}{\partial w} = w - \sum y_i \alpha_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum y_i \alpha_i = 0$$

The result is a quadratic programming problem with linear constraints

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle$$

$$\alpha_i \geq 0$$

$$\sum \alpha_i y_i = 0$$

that presents just a global maximum and can always be exactly solved efficiently. The resulting solution has the property that

$$w = \sum y_i \alpha_i x_i$$

and in fact, often most of the coefficients α_i are equal to zero. The only positive coefficients correspond to the points that lie closest to the hyperplane, and for this reason, such points go under the name of support vectors.

The final decision function can be written as

$$f(x) = \langle w, x\rangle + b = \sum y_i \alpha_i \langle x_i, x\rangle + b$$

where the index i runs only on the support vectors. In other words, if all data points other than the support vectors were removed, the algorithm would find the same solution. This property, known as *sparseness,* has many consequences, both in the implementation and in the analysis of the algorithm.

Notice also that both in the training and in the testing, the algorithm uses data-only inside inner products. This property paves the way for the use of kernel functions, which can be regarded as generalized inner products, with this algorithm (see sidebar 2).

Inner Products, Feature Spaces, and Kernels

The main idea behind kernel methods is to embed the data items into a vector space and use linear algebra and geometry to detect structure in the data. For example, one can learn classifications in the form of maximal margin hyperplanes, as described in sidebar 1. There are several reasons to embed the data into a feature space. By mapping the data into a suitable space, it is possible to transform nonlinear relations within the data into linear ones. Furthermore, the input domain does not need to be a vector space, just the embedding space needs to be.

Rather than writing down explicitly the positions of the data points within some reference frame, we make use of information about the relative positions of the points with respect to each other. In particular, we use the inner products between all pairs of vectors in the embedding space. Such information can often be obtained in a way that is independent of the dimensionality of this space. Many algorithms (for example, the maximal margin classifier described in this article) can make use of inner product information.

Figure A shows the basic idea behind support vector machines, which is to map the data into some dot product space (called the *feature space*) F using a nonlinear map $\phi: \Re^n \rightarrow F$ and perform the maximal margin algorithm in F.

Clearly, if F is high dimensional, working in the feature space can be expensive.

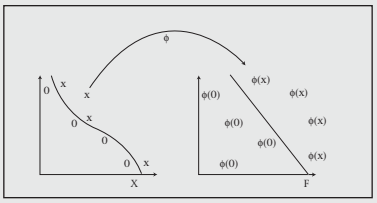


Figure A. A Nonlinear Map Is Used to Embed the Data in a Space Where Linear Relations Are Sought.

However, in sidebar 1, we saw that all we need to know to perform the maximal margin algorithm is the inner product between vectors in the feature space. This quantity can often be computed more efficiently by means of a kernel function $k(x, z) = \langle \phi(x), \phi(z) \rangle$.

This computational shortcut can be used by a large class of algorithms, including principal components analysis and ridge regression.

As an example of kernel function, consider the following map from a two-dimensional space to a three-dimensional one:

$$\phi(x_1, x_2) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$

The inner product in such a space can easily be computed without explicitly rewriting the data in the new representation.

Consider two points:

$x = (x1, x2)$

$z = (z1, z2)$

and consider the kernel function obtained by squaring their inner product:

$$\langle x, z \rangle^2 = (x_1z_1 + x_2z_2)^2 =$$
$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2 =$$
$$= \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \right\rangle =$$
$$= \langle \phi(x), \phi(z) \rangle$$

This function corresponds to the inner product between 2 three-dimensional vectors. If we had used a higher exponent, we would have virtually embedded these two vectors in a much higher-dimensional space at a very low computational cost.

More generally, we can prove that for every kernel that gives rise to a positive definite matrix $K_{ij} = k(x_i, x_j)$, we can construct a map ϕ such that $k(x, z) = \langle \phi(x), \phi(z) \rangle$.

Other examples of kernels include the Gaussian

$$k(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$$

the polynomial

$k(x, z) = (\langle x, z \rangle + 1)^d$

and kernels defined over sets such as strings and text documents. Such kernels virtually embed elements of general sets into a Euclidean space, where learning algorithms can be executed. This capability to naturally deal with general data types (for example, biosequences, images, or hypertext documents) is one of the main innovations introduced by the kernel approach.

2500 genes, each described by 79 expression values and labeled according to one of 5 functions as stated by the MIPS database. The performance was measured by the number of false positives plus twice the number of false negatives. SVMs with Gaussian kernels achieved a cost of 24 for the first class (conventional methods ranging between 28 and 41), 21 for the second class (conventional methods between 30 and 61), 17 for the third class (conventional methods between 22 and 78), 17 for the fourth class (conventional methods between 31 and 44); and 4 for the fifth class (with standard approaches ranging between 6 and 12). Other successful applications of SVMs to computational biology involve protein homology prediction, protein fold classification, and drug activity prediction.

Conclusions

The development of kernel-based learning systems in the mid-1990s represented another turning point in the history of pattern-recognition methods, comparable to the "nonlinear revolution" of the mid-1980s, when the intro-

duction of back-propagation networks and decision trees triggered a series of rapid advances, that ultimately spawned entire fields such as data mining. The present revolution comes with a new level of generalization performance, theoretical rigor, and computational efficiency. The extensive use of optimization methods, the deeper understanding of the phenomenon of overfitting from the statistical point of view, and the use of kernels as nonlinear similarity measures all represent elements of novelty, directly addressing weaknesses of the previous generation of pattern-recognition systems.

The resulting systems have rapidly become part of the toolbox of practitioners in addition to still being the object of much theoretical attention. A rich research community has emerged, and many universities and software companies participate in the research in this field, including startups that use it for mining postgenomic data. The future of this field ultimately depends on the performance of the algorithms. As long as kernel-based learning methods continue to deliver state-of-the-art performance in strategic applications such as text categorization, handwriting recognition, gene function, and cancer tissue recognition, the interest in them is bound to remain high.[2]

Kernel methods also mark an important development in AI research, demonstrating that—at least in very specific domains—a rigorous mathematical theory is not only possible but also pays off from a practical point of view. The impact of this research direction would be even larger if it could inspire neighboring fields to introduce analogous tools in their methodology. There is no doubt that a mathematical theory of intelligent systems is still far in the future, but the success of learning theory in delivering effective learning methods demonstrates that this possibility is at least not impossible.

Notes

1. www.jmlr.org.

2. Official kernel machines web site: www.kernel-machines.org.

References

Boser, B.; Guyon, I.; Vapnik, V. 1992. A Training Algorithm for Optimal Margin Classifiers. In Proceedings of the Fifth Conference on Computational Learning Theory, 144–152. New York: Association of Computing Machinery.

Brown, M.; Grundy, W.; Lin, D.; Cristianini, N.; Sugnet, C.; Furey, T.; Ares, M.; Haussler, D. 2000. Knowledge-Based Analysis of Microarray Gene Expression Data Using Support Vector Machines. *Proceedings of the National Academy of Sciences* 97:262–267.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambrige University Press. Also see www.support-vector.net.

DeCoste, D., and Schoelkopf, B. 2002. Training Invariant Support Vector Machines. *Machine Learning* 46(1–2–3): 161–190.

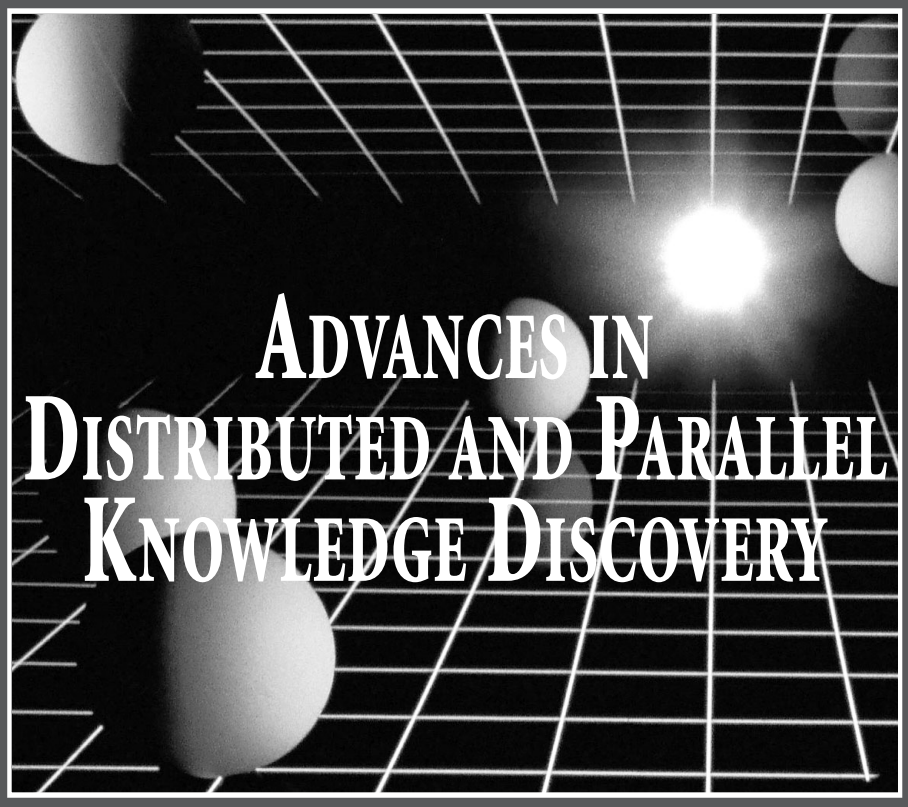
Joachims, T. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of European Conference on Machine Learning,* 137–142. Berlin: Springer-Verlag.

Schölkopf, B., and Smola, A. 2002. *Learning with Kernels*. Cambridge, Mass.: MIT Press. See also www.learning-with-kernels.org.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

Nello Cristianini works for BIOwulf, a small startup with offices in Berkeley, California, and New York City. He is a visiting lecturer at the University of California at Berkeley and honorary research associate at Royal Holloway, University of London. He is an active researcher in the area of kernel methods, was the author of a recent textbook, and was instrumental in building the primary web site of the community.

Bernhard Schölkopf works for BIOwulf, a small startup with offices in Berkeley, California, and New York City. He is also director of the Max-Planck-Institute for Biological Cybernetics, Tübingen, Germany. He is an active researcher in the area of kernel methods, is the author of a recent textbook, and was instrumental in building the primary web site for the community.



Edited by Hillol Kargupta and Philip Chan

Knowledge discovery and data mining (KDD) deals with the problem of extracting interesting associations, classifiers, clusters, and other patterns from data. The emergence of network-based distributed computing environments has introduced an important new dimension to this problem—distributed sources of data. Distributed knowledge discovery (DKD) works with the merger of communication and computation by analyzing data in a distributed fashion. This technology is particularly useful for large heterogeneous distributed environments such as the Internet, intranets, mobile computing environments, and sensor networks. When the datasets are large, scaling up the speed of the KDD process is crucial. Parallel knowledge discovery (PKD) techniques addresses this problem by using high-performance multi-processor machines. This book presents introductions to DKD and PKD, extensive reviews of the field, and state-of-the-art techniques.

PUBLISHED BY THE AAAI PRESS / COPUBLISHED BY THE MIT PRESS
Five Cambridge Center, Cambridge, Massachusetts 02142 USA

http://mitpress.edu/ • 800-405-1619
ISBN 0-262-61155-4, 472 pp., illus., bibliography, index